```
% Limpieza del entorno y preparación de la sesión
clear all; close all; clc;
tic;  % Inicia el cronometraje

%---------------------------------------------------------------------------
% Declaración de variables simbólicas
%---------------------------------------------------------------------------
% Variables de ángulos, velocidades y aceleraciones (funciones de t)
syms th1(t) th2(t) t              % Ángulos de cada articulación
syms th1p(t) th2p(t)              % Velocidades articulares
syms th1pp(t) th2pp(t)            % Aceleraciones articulares

% Parámetros físicos: masas, momentos de inercia y longitudes
syms m1 m2 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2
syms l1 l2 lc1 lc2               % l: longitud del eslabón, lc: distancia al
centro de masa
syms pi g a cero                 % Constantes físicas y auxiliares

%---------------------------------------------------------------------------
% Vectores articulares: coordenadas, velocidades y aceleraciones
%---------------------------------------------------------------------------
Q = [th1; th2];                 % Vector de coordenadas articulares
Qp = [th1p; th2p];              % Vector de velocidades articulares
Qpp = [th1pp; th2pp];            % Vector de aceleraciones articulares

%---------------------------------------------------------------------------
% Configuración del robot: Robot rotacional de 2 GDL
%   0 -> junta rotacional, 1 -> junta prismática
%---------------------------------------------------------------------------
RP = [0 0];                     % Ambas juntas son rotacionales
GDL = size(RP,2);               % Número de grados de libertad
GDL_str = num2str(GDL);         % Conversión a cadena (si fuera necesario)

%---------------------------------------------------------------------------
% Cinemática directa
%---------------------------------------------------------------------------
% Articulación 1: posición y orientación respecto al marco base
P(:,:,1) = [l1*cos(th1); l1*sin(th1); 0];   % Posición de la articulación 1
R(:,:,1) = [cos(th1) -sin(th1) 0;
            sin(th1)  cos(th1) 0;
            0         0        1];         % Matriz de rotación de la junta 1

% Articulación 2: posición y orientación respecto al eslabón 1
P(:,:,2) = [l2*cos(th2); l2*sin(th2); 0];   % Posición de la articulación 2
R(:,:,2) = [cos(th2) -sin(th2) 0;
            sin(th2)  cos(th2) 0;
            0         0        1];         % Matriz de rotación de la junta 2

%---------------------------------------------------------------------------
% Construcción de las matrices de transformación homogénea
```

```
%-------------------------------------------------------------------------
Vector_Zeros = zeros(1, 3);   % Vector para completar la fila [0 0 0 1]

% Inicializamos las matrices locales y globales
A(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
T(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
PO(:,:,GDL) = P(:,:,GDL);     % Posición final del eslabón
RO(:,:,GDL) = R(:,:,GDL);     % Orientación final del eslabón

% Se calculan las transformaciones globales para cada eslabón
for i = 1:GDL
    A(:,:,i) = simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
    try
       T(:,:,i) = T(:,:,i-1) * A(:,:,i);
    catch
       T(:,:,i) = A(:,:,i);
    end
    T(:,:,i) = simplify(T(:,:,i));
    RO(:,:,i) = T(1:3,1:3,i);
    PO(:,:,i) = T(1:3,4,i);
end

%-------------------------------------------------------------------------
% Cálculo de Jacobianos y Velocidades
%-------------------------------------------------------------------------
% Para juntas rotacionales, se utiliza el producto cruzado para el Jacobiano

% --- Eslabón 2 ---
% Inicializamos el Jacobiano lineal y angular para el eslabón 2
Jv_a2(:,GDL) = PO(:,:,GDL);
Jw_a2(:,GDL) = PO(:,:,GDL);

for k = 1:GDL
    if RP(k) == 0    % Junta rotacional
        try
            % Se calcula el producto cruzado entre el eje de rotación y el
vector diferencia
            Jv_a2(:,k) = cross(RO(:,3,k-1), PO(:,:,GDL) - PO(:,:,k-1));
            Jw_a2(:,k) = RO(:,3,k-1);
        catch
            % En caso de que no exista matriz previa se usa el eje z
            Jv_a2(:,k) = cross([0;0;1], PO(:,:,GDL));
            Jw_a2(:,k) = [0;0;1];
        end
    else   % (No aplica en este robot rotacional)
        try
            Jv_a2(:,k) = RO(:,3,k-1);
        catch
            Jv_a2(:,k) = [0;0;1];
        end
```

```
        Jw_a2(:,k) = [0;0;0];
    end
end

Jv_a2 = simplify(Jv_a2);
Jw_a2 = simplify(Jw_a2);
Jac2 = [Jv_a2; Jw_a2];
Jacobiano2 = simplify(Jac2);

% Se evalúa el vector de velocidades como función de t
Qp = Qp(t);

disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 2');
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 2
```

```
V2 = simplify(Jv_a2 * Qp);
pretty(V2);
```

```
/ - th1p(t) (l1 sin(th1(t)) + l2 #1) - l2 #1 th2p(t) \
|                                                    |
|  th1p(t) (l1 cos(th1(t)) + l2 #2) + l2 #2 th2p(t)  |
|                                                    |
\                          0                         /

where

   #1 == sin(th1(t) + th2(t))

   #2 == cos(th1(t) + th2(t))
```

```

disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón
2');
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 2
```

```
W2 = simplify(Jw_a2 * Qp);
pretty(W2);
```

```
/         0         \
|                   |
|         0         |
|                   |
\ th1p(t) + th2p(t) /
```

```


% --- Eslabón 1 ---
% Para el eslabón 1 se considera solo la primera junta
Jv_a1(:,GDL-1) = PO(:,:,GDL-1);
Jw_a1(:,GDL-1) = PO(:,:,GDL-1);

for k = 1:GDL-1
    if RP(k) == 0    % Junta rotacional
        try
            Jv_a1(:,k) = cross(RO(:,3,k-1), PO(:,:,GDL-1) - PO(:,:,k-1));
```

```
            Jw_a1(:,k) = RO(:,3,k-1);
        catch
            Jv_a1(:,k) = cross([0;0;1], PO(:,:,GDL-1));
            Jw_a1(:,k) = [0;0;1];
        end
    else
        try
            Jv_a1(:,k) = RO(:,3,k-1);
        catch
            Jv_a1(:,k) = [0;0;1];
        end
        Jw_a1(:,k) = [0;0;0];
    end
end

Jv_a1 = simplify(Jv_a1);
Jw_a1 = simplify(Jw_a1);
Jac1 = [Jv_a1; Jw_a1];
Jacobiano1 = simplify(Jac1);

disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 1');
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 1
```

```
V1 = simplify(Jv_a1 * Qp(1:1));
pretty(V1);
```

```
/ -l1 sin(th1(t)) th1p(t) \
|                         |
|  l1 cos(th1(t)) th1p(t) |
|                         |
\            0            /
```

```
disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 
1');
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 1
```

```
W1 = simplify(Jw_a1 * Qp(1:1));
pretty(W1);
```

```
/    0    \
|         |
|    0    |
|         |
\ th1p(t) /
```

```
%--------------------------------------------------------------------------
% Cálculo de la Energía Cinética
%--------------------------------------------------------------------------
% Se sustituyen las longitudes totales por las distancias al centro de masa 
para cada eslabón
```

```
P01 = subs(P(:,:,1), l1, lc1);   % Eslabón 1
P12 = subs(P(:,:,2), l2, lc2);   % Eslabón 2

% Se definen las matrices de inercia para cada eslabón
I1 = [Ixx1 0 0; 0 Iyy1 0; 0 0 Izz1];
I2 = [Ixx2 0 0; 0 Iyy2 0; 0 0 Izz2];

% Energía Cinética de cada eslabón
% Para cada eslabón se calcula la velocidad total sumando la contribución
del centro de masa

% Eslabón 1
V1_Total = V1 + cross(W1, P01);
K1 = (1/2 * m1 * (V1_Total)') * (V1_Total) + (1/2 * W1') * (I1 * W1);
disp('Energía Cinética en el Eslabón 1');
```

```
Energía Cinética en el Eslabón 1
```

```
K1 = simplify(K1);
pretty(K1);
```

```
              2               2     ______                                    2           2
Izz1 |th1p(t)|    m1 |th1p(t)|  cos(th1(t) - th1(t)) (l1 + lc1) (lc1 |l1|  + l1 |lc1| )
--------------- + -------------------------------------------------------------------
       2                                          2 l1 lc1
```

```
% Eslabón 2
V2_Total = V2 + cross(W2, P12);
K2 = (1/2 * m2 * (V2_Total)') * (V2_Total) + (1/2 * W2') * (I2 * W2);
disp('Energía Cinética en el Eslabón 2');
```

```
Energía Cinética en el Eslabón 2
```

```
K2 = simplify(K2);
pretty(K2);
```

```
                                                                                   _______           __
m2 (th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + lc2 cos(th2(t)) #1) (th1p(t) (cos(#3) l2
---------------------------------------------------------------------------------------------------------
                                                                                 2

where

   #1 == th1p(t) + th2p(t)

         _______   _______
   #2 == th1p(t) + th2p(t)

         ______   ______
   #3 == th1(t) + th2(t)

   #4 == th1(t) + th2(t)
```

```
% Energía Cinética Total del robot
K_Total = simplify(K1 + K2);
```

```
disp('Energía Cinética Total');
```

```
Energía Cinética Total
```

```
pretty(K_Total);
```

```
                                                                                                  _______
Izz1 #5   m2 (th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + lc2 cos(th2(t)) #1) (th1p(t) (c
------- + ---------------------------------------------------------------------------------------------
   2                                                                                         2

where

   #1 == th1p(t) + th2p(t)

         _______   _______
   #2 == th1p(t) + th2p(t)

         ______   ______
   #3 == th1(t) + th2(t)

   #4 == th1(t) + th2(t)

                       2
   #5 == |th1p(t)|
```

```
%----------------------------------------------------------------------------
% Cálculo de la Energía Potencial
%----------------------------------------------------------------------------
% Se definen las alturas relevantes para cada eslabón (según la orientación
del sistema)
h1 = P01(2);  % Altura del eslabón 1 (eje y)
h2 = P12(2);  % Altura del eslabón 2 (eje y)

U1 = m1 * g * h1;   % Energía potencial del eslabón 1
U2 = m2 * g * h2;   % Energía potencial del eslabón 2

% Energía potencial total
U_Total = U1 + U2;

%----------------------------------------------------------------------------
% Cálculo del Lagrangiano y Modelo de Energía
%----------------------------------------------------------------------------
Lagrangiano = simplify(K_Total - U_Total);
disp('Lagrangiano L = K - U');
```

```
Lagrangiano L = K - U
```

```
pretty(Lagrangiano);
```

```
                                                                                                  _______
Izz1 #5   m2 (th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + lc2 cos(th2(t)) #1) (th1p(t) (c
------- + ---------------------------------------------------------------------------------------------
   2                                                                                         2

where
```

```
   #1 == th1p(t) + th2p(t)

         _______   _______
   #2 == th1p(t) + th2p(t)

         ______   ______
   #3 == th1(t) + th2(t)

   #4 == th1(t) + th2(t)

                        2
   #5 == |th1p(t)|
```

```
H = simplify(K_Total + U_Total);
disp('Función de Energía Total H = K + U');
```

```
Función de Energía Total H = K + U
```

```
pretty(H);
```

```
                                                                                                        _______
Izz1 #5   m2 (th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + lc2 cos(th2(t)) #1) (th1p(t) (c
------- + ------------------------------------------------------------------------------------------------
   2                                                                                              2

where

   #1 == th1p(t) + th2p(t)

         _______   _______
   #2 == th1p(t) + th2p(t)

         ______   ______
   #3 == th1(t) + th2(t)

   #4 == th1(t) + th2(t)

                        2
   #5 == |th1p(t)|
```

```
toc;  % Finaliza el cronometraje
```

```
Elapsed time is 3.035629 seconds.
```

Obtención de la Energía Cinética Total en un Robot Rotacional de 2 GDL

Introducción

Este reporte describe el procedimiento para obtener la energía cinética total en un robot rotacional de 2 grados de libertad (2 GDL). El robot cuenta con dos juntas rotacionales, por lo que el movimiento se caracteriza mediante ángulos, velocidades angulares y aceleraciones angulares.

Objetivos

- Definir la cinemática del robot: Determinar la posición y orientación de cada eslabón en función de los ángulos de las juntas.
- Calcular los Jacobianos y las velocidades: Relacionar las velocidades articulares con las velocidades lineales y angulares de cada eslabón.
- Obtener la energía cinética total: Incluir las contribuciones traslacional y rotacional.
- Establecer el Lagrangiano y la función de energía total: Elementos clave para el análisis dinámico y el diseño de control.

Metodología

Procedimiento1. Preparación del Entorno y Declaración de Variables

- Descripción:Se limpia el entorno de MATLAB y se definen las variables simbólicas para los ángulos (θ1 y θ2), sus derivadas y los parámetros físicos (masas, inercia, longitudes y distancias al centro de masa).

2. Definición de la Cinemática Directa

- **Descripción:**Se establece la posición y orientación de cada eslabón del robot mediante funciones trigonométricas y matrices de rotación.
- **Pasos:**
- Definición de P(:,:,1) y R(:,:,1).
- Definición de P(:,:,2) y R(:,:,2).

3. Construcción de las Matrices de Transformación Homogénea

- **Descripción:**Se combinan las posiciones y orientaciones en matrices homogéneas que permiten transformar las coordenadas del marco base al marco de cada eslabón.
- **Pasos:**
- Construcción de las matrices locales A(:,:,i).
- Cálculo de las matrices globales T(:,:,i) mediante la multiplicación secuencial.
- Extracción de la posición PO y la orientación RO.

4. Cálculo del Jacobiano y de las Velocidades

- **Descripción:**Se obtienen los Jacobianos analíticos para cada eslabón mediante el producto cruzado del eje de rotación con la diferencia de posiciones. Luego, se multiplican los Jacobianos por el vector de velocidades articulares para calcular:
- La velocidad lineal V.
- La velocidad angular W.

- **Pasos:**
- Cálculo del Jacobiano para el eslabón 2 usando ambas juntas.

- Cálculo del Jacobiano para el eslabón 1 utilizando la primera junta.
- Obtención de V2,W2 y V1,W1.

5. Cálculo de la Energía Cinética

- **Descripción:**La energía cinética se obtiene considerando dos componentes:
- **Traslacional:** Relacionada a la velocidad total del centro de masa. Se calcula sustituyendo la longitud total por la distancia al centro de masa.
- **Rotacional:** Dependiente de la velocidad angular y la distribución de masa (matriz de inercia).
- **Pasos:**
- Sustitución de li por lci en la posición para obtener P01 y P12.
- Cálculo de la velocidad total VTotal=V+cross(W,Pcm) para cada eslabón.
- Cálculo de la energía cinética
- Suma de las energías cinéticas parciales para obtener KTotal

6. Cálculo de la Energía Potencial y del Lagrangiano

- **Descripción:**La energía potencial se calcula en función de la altura del centro de masa (en este caso, se toma la componente y de la posición). El Lagrangiano se define como la diferencia entre la energía cinética y la potencial.
- **Pasos:**
- Determinación de las alturas h1 y h2 para cada eslabón.
- Cálculo de U=mgh para cada eslabón y suma para UTotal.
- Cálculo del Lagrangiano L=KTotal − UTotal y de la función de energía total H=KTotal+UTotal

Conclusión

El procedimiento integra la definición de la cinemática y dinámica de un robot rotacional de 2 GDL. A partir de la formulación se obtienen los Jacobianos y velocidades, que permiten calcular las energías cinética y potencial. Este análisis es fundamental para el desarrollo de modelos de control y simulación de robots en entornos dinámicos.